NormalGLM Assigment

Nicholas Cahill

November 21, 2018

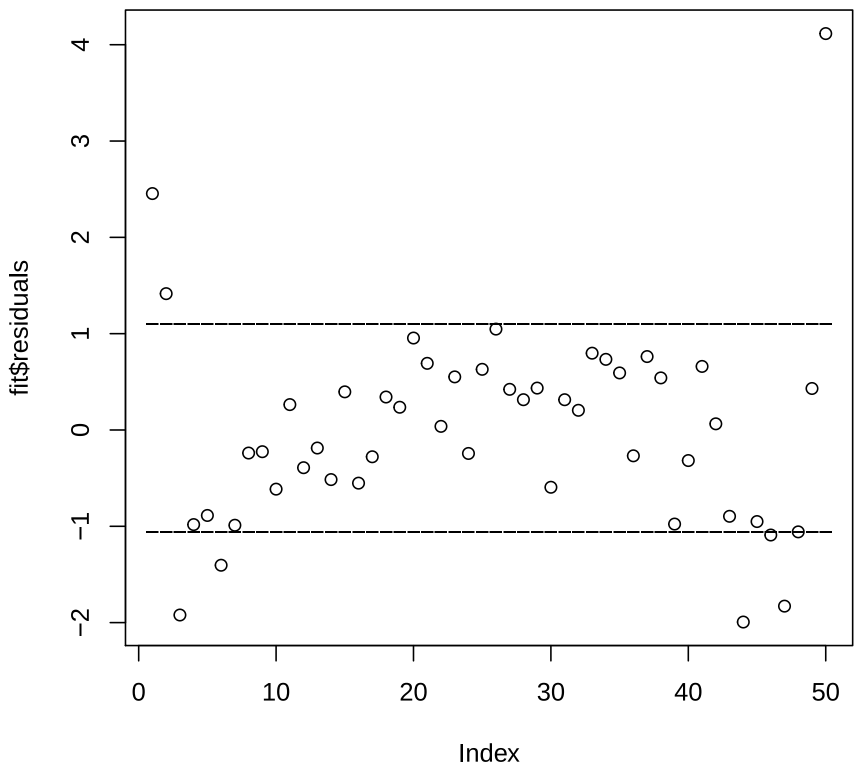
Part a

After loading our data and normal linear model, we'll look at the significance of the quadratic term.

```
> load("~/Downloads/rstuff/Normal_GLM.Rdata")
> xsq = x^2
> fit = lm(y~x+xsq)
> summary(fit)

Call:
lm(formula = y ~ x + xsq)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9940 -0.6093  0.0508  0.5487  4.1157

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.4348     0.5395   19.34  < 2e-16 ***
x           -27.1207     2.2597  -12.00 6.46e-16 ***
xsq          27.5193     2.0916   13.16  < 2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 1.08 on 47 degrees of freedom
Multiple R-squared:  0.793,	Adjusted R-squared:  0.7842
F-statistic: 90.01 on 2 and 47 DF,  p-value: < 2.2e-16
```

Looking at the results of the automatic Z-test, we can conclude that the quadratic term is significant.

Let's look at the residuals:



It looks good, since there are lots of points within the +/- 1 stdev bars... But actually there are too many points within the bars. We should only have 70% between the bars! 30% of the points SHOULD be outside the bars, but only 7 of 50 (14%) points are outside the bars. Furthermore, all of the points outside are for small or large x-values.

Part b

```
> b_1 = fit$coefficients/sigsq
> b_2 = c(1/sigsq,0,0)
> b = c(b_1,b_2)
> bHlin = b
```

Looking at the code, we see that $\beta_1 X1 = \eta_1$ and $\beta_2 X2 = \eta_2$. This would be useful if we decided to use different covariates for η_1 and η_2, but for our purposes, $X1$ and $X2$ are the same.

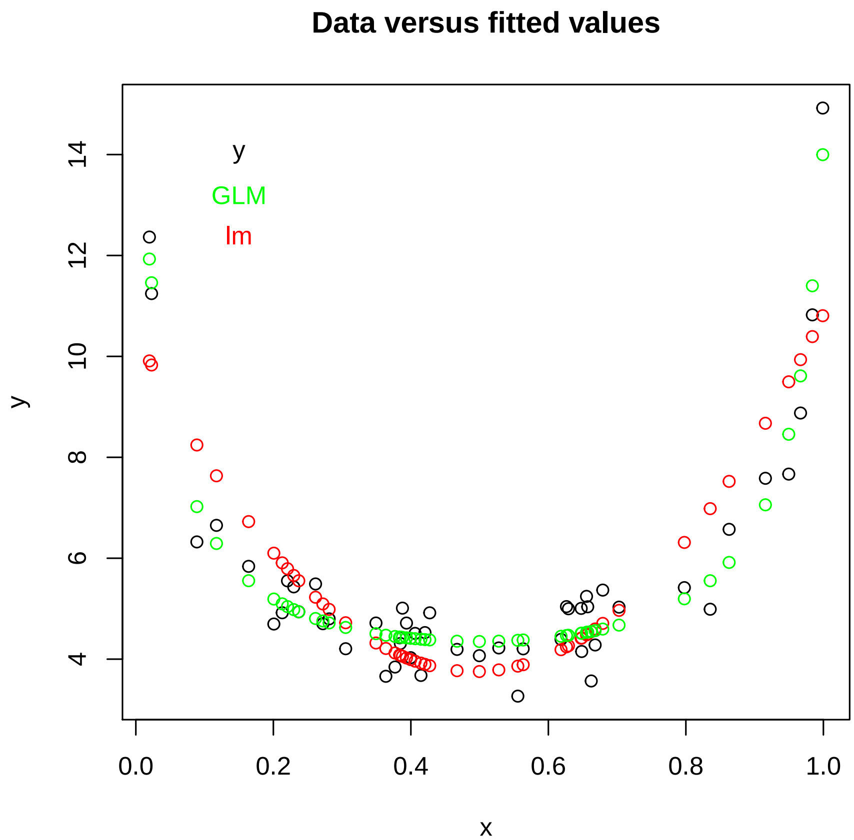
```
> lik(b_1,b_2,y,model.matrix(fit),model.matrix(fit))

[1] -73.27447

> Dlik(b_1,b_2,y,model.matrix(fit),model.matrix(fit))

  (Intercept)             x           xsq          <NA>          <NA>
 2.273737e-13  8.526513e-14  2.842171e-14  1.748165e+00 -1.213790e+00
         <NA>
-8.860241e+00
```

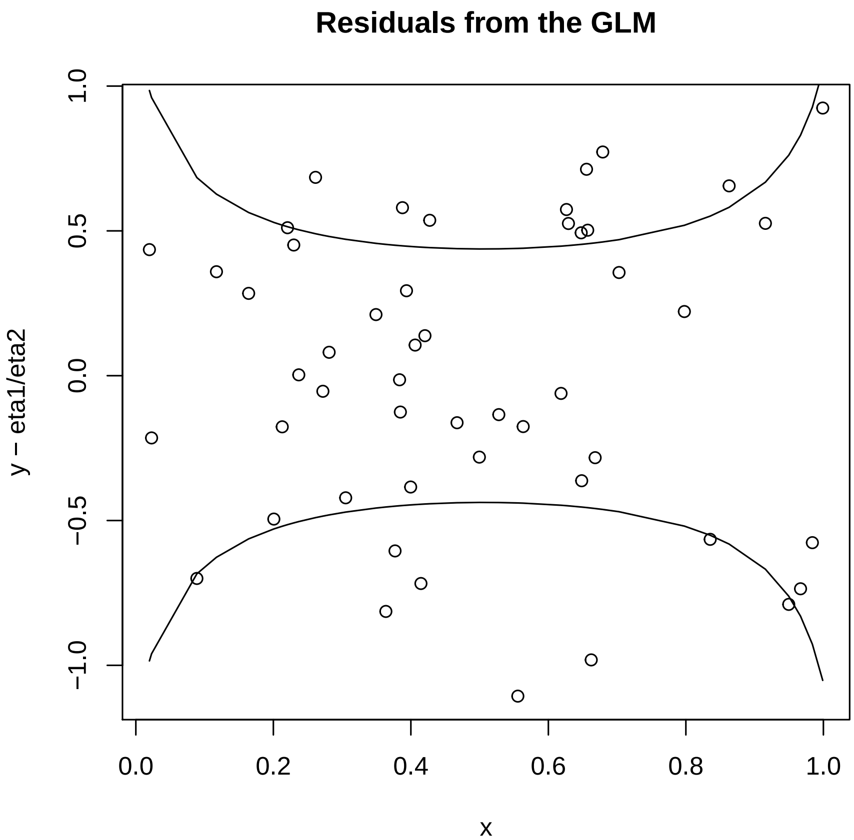
Looking at the gradient, we can see that it's very very small in the β_1 directions. This makes sense, because we just optimized those guys with *lm*... So the gradient is telling us that to improve log likelihood right now, we should change the β_2 coefficients a lot and not change the β_1 coeffficients much. First-order methods are shortsighted, and obviously the β_1 will have to change as soon as we start messing with β_2...

Part c

Source code for fitting the model is in GLMfragmentEXTRA.R, for now the fits were computed and saved to be used for this report.

```
> b = bH2
> eta1 = model.matrix(fit)%*%b[1:3] #\eta_{1,i} = X_i*\beta_1
> eta2 = model.matrix(fit)%*%b[4:6] #\eta_{2,i} = X_i*\beta_2
```


Data versus fitted values
y
GLM
lm
14
12
10
8
6
4
0.0
0.2
0.4
0.6
0.8
1.0
x
y



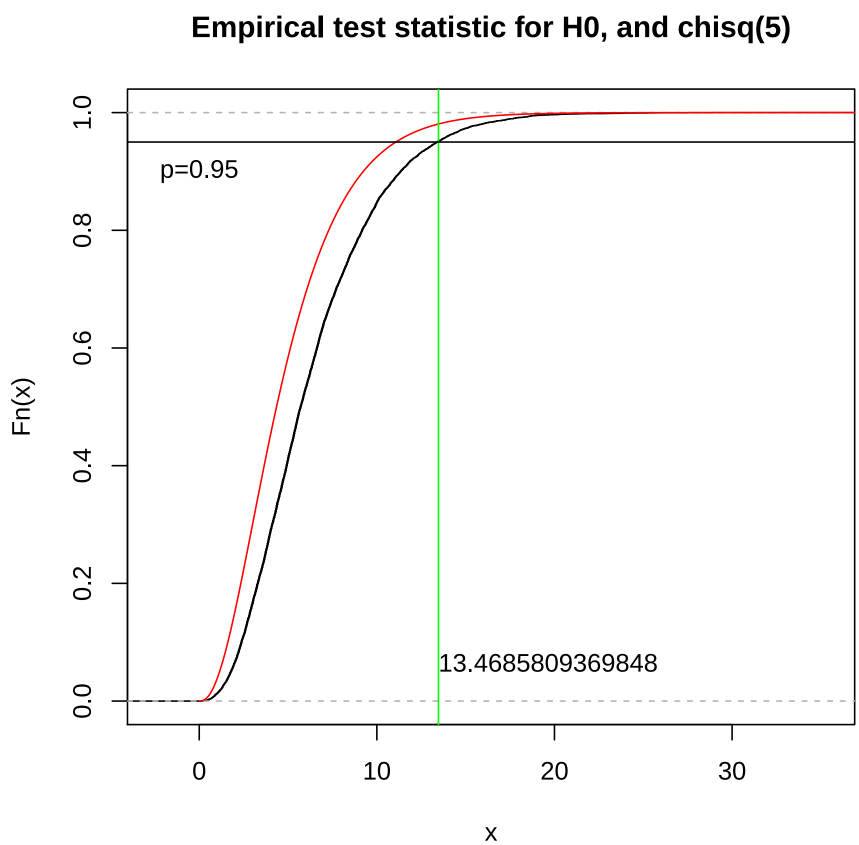
Part d

To test the hypotheses given, we fit new models and compute their log-liklihood.

Then $-2(ll(H_0) - ll(H))$ should be distributed according to a chi-squared distribution, with degrees of freedom equal to the dimension of the space of models satisfying the hypotheses.

First hypothesis, $\eta 1$ has quadratic term equal to zero. This gives us $q = 5$

```
> bH0

 (Intercept)            x
14.912840400  0.002675046  0.000000000  1.118181193  9.949502227 -9.914567860
```

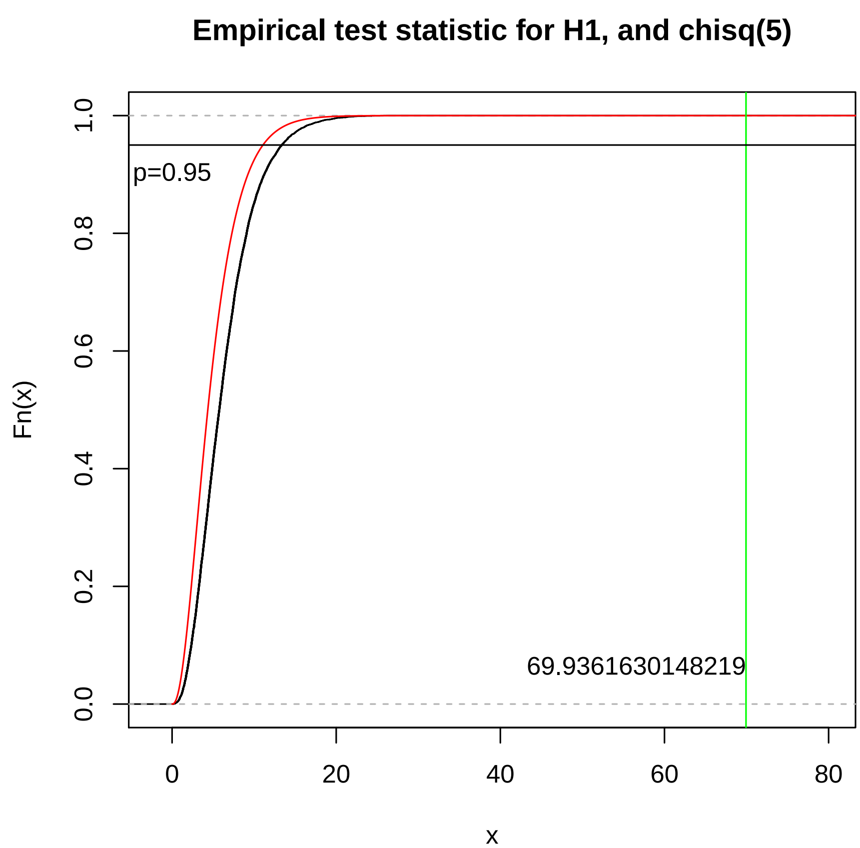



```
> mean(BBH0 > 2*(l(bH2)-l(bH0))) # Empirical p-value
[1] 0.0484
> pchisq(2*(l(bH2)-l(bH0)),5,lower.tail = FALSE) # Chi squared p-value
[1] 0.01936184
```

The chi-squared test would tell us soundly to reject this hypothesis, but the empirical distribution puts us right on the line! I wouldn't reject.

Next hypothesis, $\eta 2$ has quadratic term equal to zero. This gives us $q = 5$ again

```
> bH1

(Intercept)           x         xsq
  9.9645452 -25.5552986  25.0106108   0.9747917  -0.1166870   0.0000000
```



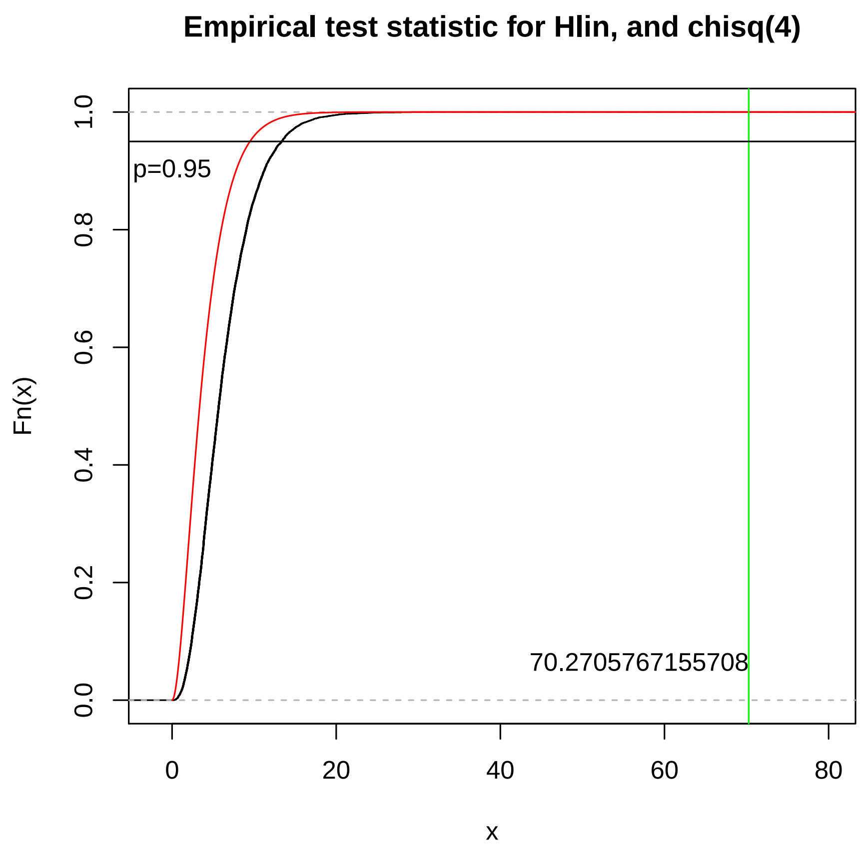

```
> mean(BBH1 > 2*(l(bH2)-l(bH1))) # Empirical p-value

[1] 0

> pchisq(2*(l(bH2)-l(bH1)),5,lower.tail = FALSE) # Chi squared p-value

[1] 1.056622e-13
```

The variable $BBH1$ has 10,000 simulations of the test statistic from the hypothesis we are testing. The test above shows that our real statistic is more extreme than all of them, so p<0.0001

In either case, we can safely reject this hypothesis.

Next hypothesis, equal variances. This gives us $q = 4$, since we are losing all by the intercept for $\eta 2$

```
> bHlin

(Intercept)           x         xsq
  8.9534714 -23.2707348  23.6127241    0.8580424    0.0000000    0.0000000
```




```
> mean(BBlin > 2*(l(bH2)-l(bHlin))) # Empirical p-value

[1] 0

> pchisq(2*(l(bH2)-l(bHlin)),4,lower.tail = FALSE) # Chi squared p-value

[1] 1.990076e-14
```

The variable $BBlin$ has 10,000 simulations of the test statistic, so the test above shows that once again our real statistic is more extreme than all of the simulated statistics, giving p<0.0001

So, once again, we can reject this hypothesis.

e

For completeness, here's the full code I used to run my simulations. The vector $bH2$ contains the fitted GLM $beta$ values.

```
> yt = (c(1.0,0.75,0.75^2)%*%bH2[1:3])/(c(1.0,0.75,0.75^2)%*%bH2[4:6])
> N=1000
> ysims1 = vector("numeric",N)
> sigsims = vector("numeric",N)
> ysims2 = vector("numeric",N)
> for(i in 1:N){
+   #First, simulate data:
+   #N(m,s) = m + \sqrt(s)N(0,1)
+   n  = rnorm(50)
+   sim = eta1/eta2 + sqrt(1/eta2)*n
+   #Set up the gradient functions so we can fit the GLM
+   D2ls=function(b)  {D2lik(b[1:3],b[4:6],sim,model.matrix(fit),model.matrix(fit))}
+   Dls=function(b)  {Dlik(b[1:3],b[4:6],sim,model.matrix(fit),model.matrix(fit))}
+   ls=function(b)  {lik(b[1:3],b[4:6],sim,model.matrix(fit),model.matrix(fit))}
+   h_ms=function(b){-solve(D2ls(b))%*%Dls(b)}
+   #Intialize values for fitting
+   b=c(b_1,b_2)
+   h=h_ms(b)
+   del=1
+   #Fit with Newton-Raphson
+   while(del>10^-35){
+     while(is.nan(ls(b+h))||ls(b+h)<ls(b)){
+       h = h/2
+     }
+     del = ls(b+h)-ls(b)
+     b = b+h
+     h = h_ms(b)
+   }
+   bH0=b
+   #Collect the just-fitted model's prediction at x=0.75
+   ysims1[i] = (c(1.0,0.75,0.75^2)%*%bH0[1:3])/(c(1.0,0.75,0.75^2)%*%bH0[4:6])
+   sigsims[i]=(c(1.0,0.75,0.75^2))%*%(1/bH0[4:6])
+   #and the standard linear model's prediction, too
+   ysims2[i] = c(1.0,0.75,0.75^2)%*%lm(sim~x+xsq)$coefficients
+ }
> mean(ysims1-yt)
```

```
[1] 0.01148324

> sd(ysims1-yt)

[1] 0.1163764

> #~0.016 and ~0.122
> mean(ysims2-yt)

[1] 0.6947669

> sd(ysims2-yt)

[1] 0.1268197

> #~0.695 and ~0.127
```

The linear model is highly biased, but has roughly the same variance. Interpreting these standard deviation numbers along with the fact that the 'true' y is 4.883219, we have a 68% chance of being less than 2.5% off, and a 95% chance of being less than 5% off. Not too bad!